머신러닝 & 딥러닝 6

AI 학술동아리 <MLP>

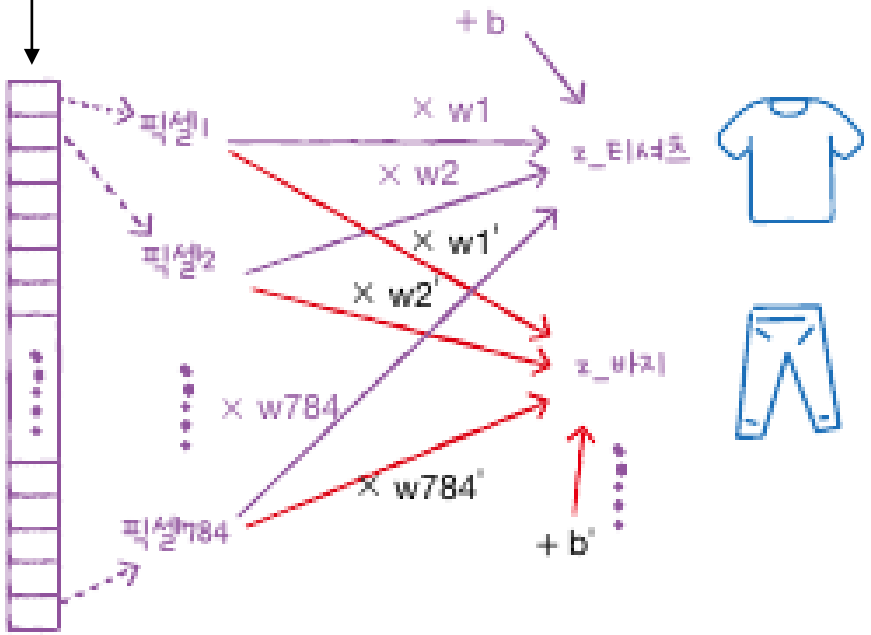
- Index

- 이미지 데이터 정규화 하기

이미지의 픽셀은 0~255 사이의 정숫값을 가짐
-> 이 경우 보통 255로 나누어 0~1 사이의 값으로 정규화
(표준화는 아니지만 양수 값으로 이루어진 이미지를 전처리할 때 널리 사용하는 방법)

0. Logistic Regression을 써서 이미지 데이터 Classification하기

2차원 배열인 이미지 데이터를 1차원 배열로 변경

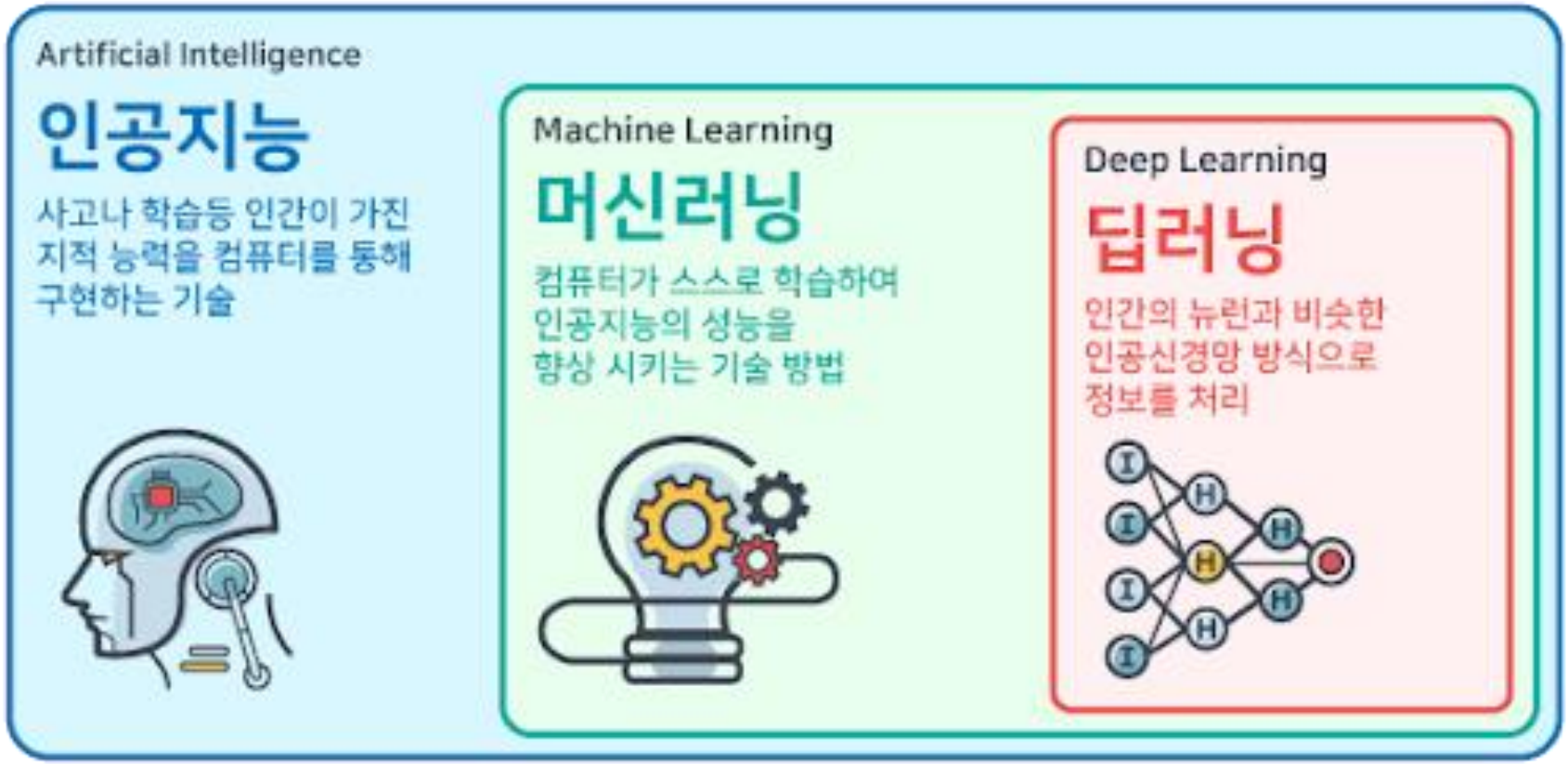


$$z\_티셔츠 = w1 \times (픽셀1) + w2 \times (픽셀2) + \ldots + w784 \times (픽셀784) + b$$

$$z\_바지 = w1' \times (픽셀1) + w2' \times (픽셀2) + \ldots + w784' \times (픽셀784) + b'$$

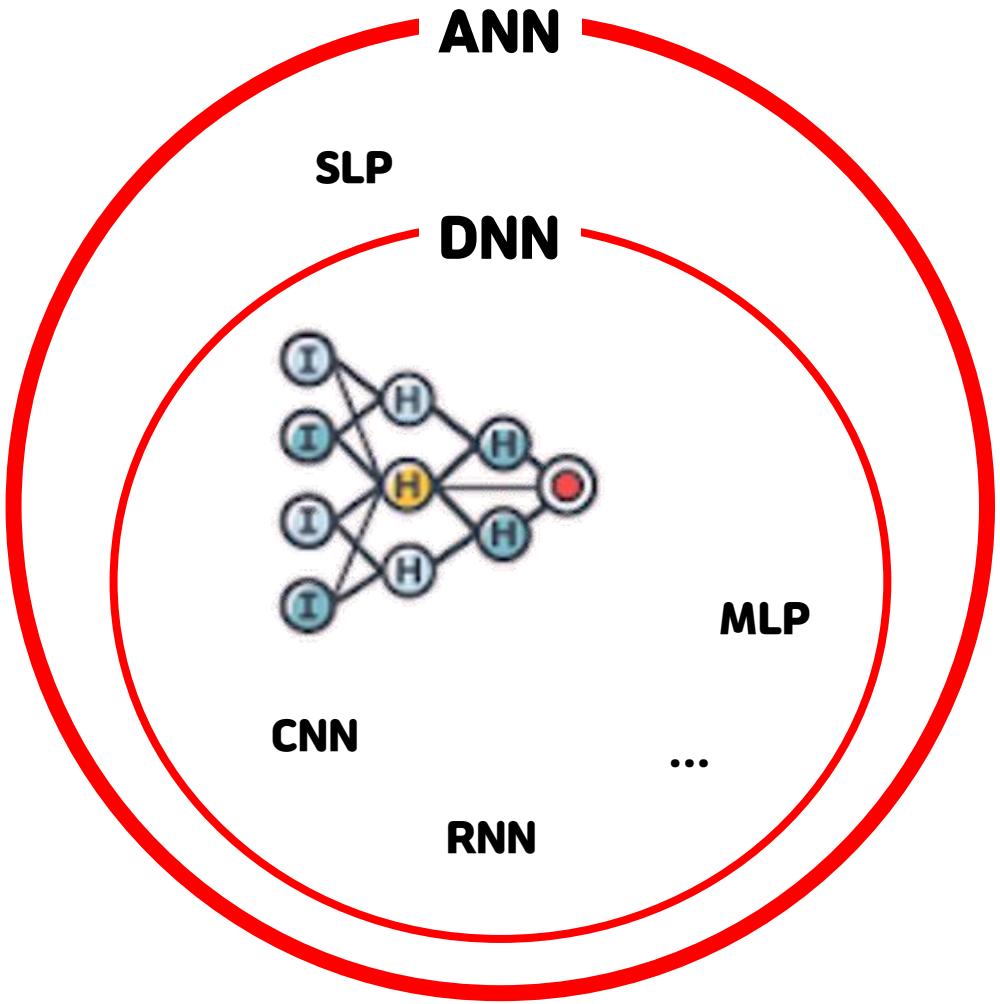
=> z값을 softmax 함수에 통과시켜 각 클래스에 대한 확률 얻기

- Artificial Intelligence(인공지능)



DL 라이브러리는 다른 ML 라이브러리와 다르게 GPU사용해서 ANN훈련
- GPU는 벡터와 행렬 연산에 매우 최적화

1. Artificial Neural Network(인공 신경망)



ANN을 줄여서 NN(Neural Network)라고도 함

ANN을 DL이라고도 함

ANN(Artificial Neural Network) = 인공 신경망

DNN(Deep Neural Network) = 심층 신경망
= Deep Learning

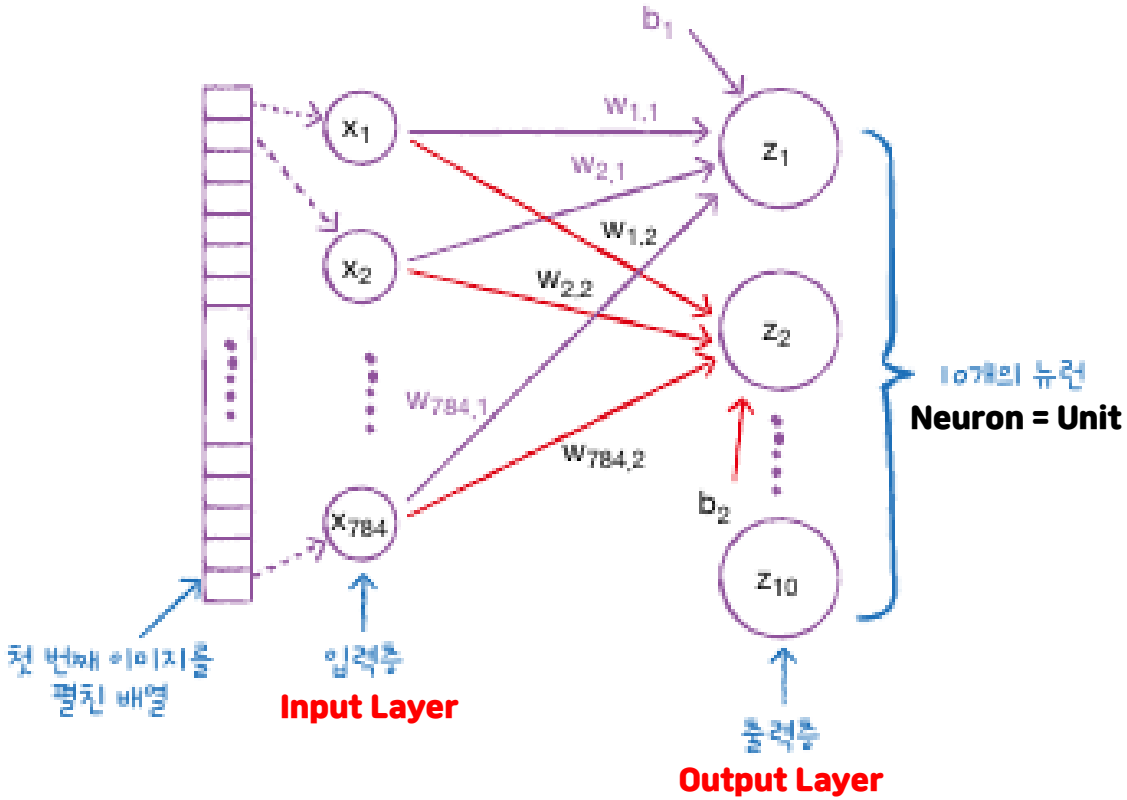
-> Layer가 여러개

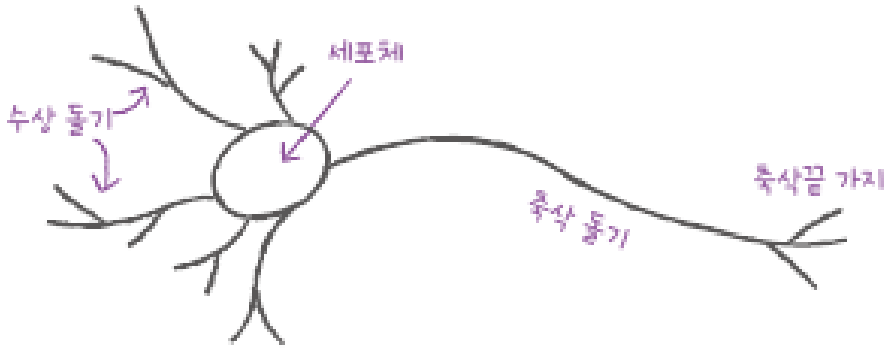
SLP(Single Layer Perceptron) = 단층 퍼셉트론

MLP(Multi Layer Perceptron) = 다층 퍼셉트론

1. ANN

ANN(Artificial Neural Network) : 생물학적 뉴런에서 영감을 얻어 만들어진 새로운 종류의 머신러닝 알고리즘





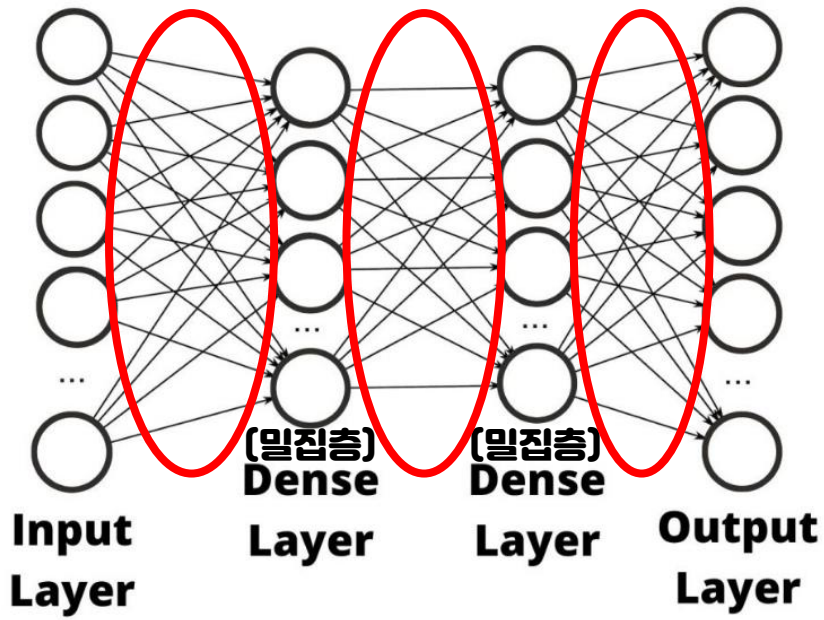
수상 돌기 : Input Node
세포체 : Linear Function + Activation Function
축삭 돌기 : Output

Linear Function : Linear Regression, Logistic Regression 등
Activation Function : sigmoid function, softmax function 등

모든 동그라미를 각각 Node(노드)라고 칭함

1. ANN

DL에서는 cross validation을 잘 사용하지 않고, validation set를 별도로 덜어내어 사용
- DL 분야의 데이터셋은 충분히 크기 때문에 검증 점수가 안정적
- cross validation을 하기에는 훈련 시간이 너무 오래 걸림



Input Layer와 Output Layer 사이의 Layer를
숨겨졌단 의미에서 Hidden Layer(은닉층)라고 부름

Hidden Layer가 항상 FC Layer는 아닐 수 있음
- 몇 개 연결이 끊긴 Hidden Layer도 존재

Hidden Layer ⊃ FC Layer

Dense Layer = Fully Connected Layer(FC Layer) (완전 연결층)

Remind! Sigmoid, Softmax

$$f(x) = \frac{1}{1+e^{-x}}$$

로지스틱 함수

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^{z}}{e^{z}+1}$$

binary classification

sigmoid 시그모이드

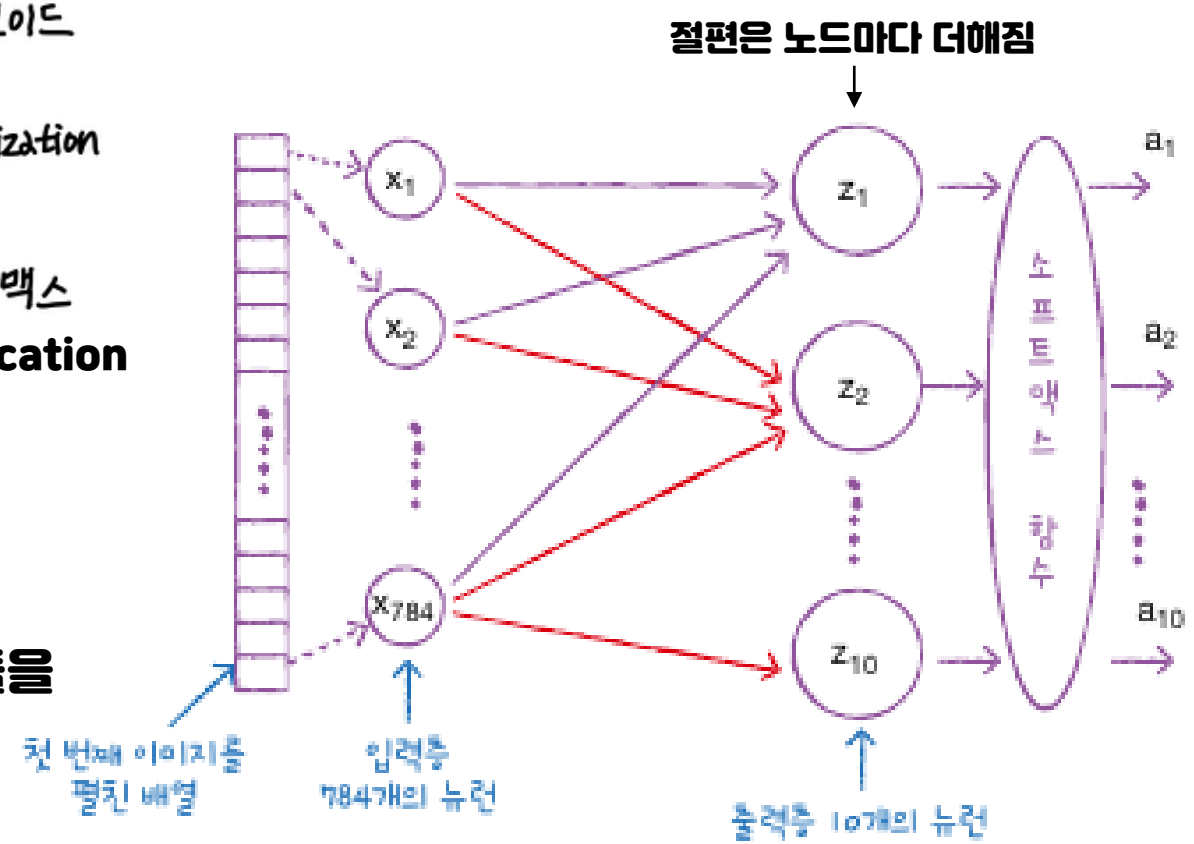
일반화 generalization

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$$

Softmax 소프트맥스

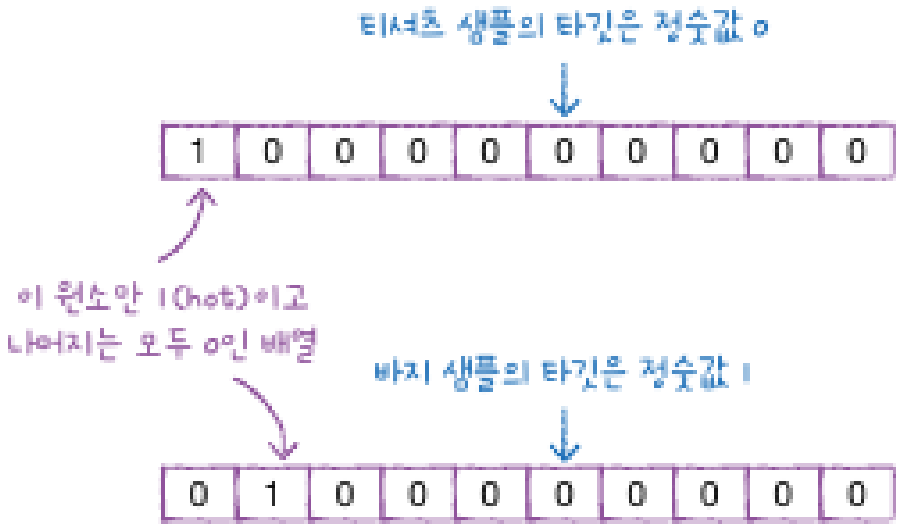
multiclass classification

z값들이 해당 함수를 거쳐 각 class의 확률로 나옴



Node의 선형 방정식 계산 결과에 적용되는 이와같은 함수들을 Activation Function(활성화 함수)라고 부름

- One-hot encoding(원-핫 인코딩)



티셔츠 : 0, 바지 : 1, 스웨터 : 2, … 일 때,
사진의 정답값을 0, 1, 2, … 로 나타내는 것이 아니라 왼쪽과 같이
해당 클래스만 1이고 나머지는 모두 0인 배열로 만드는 것을
One-hot encoding이라고 함

- ML Library(대표적)

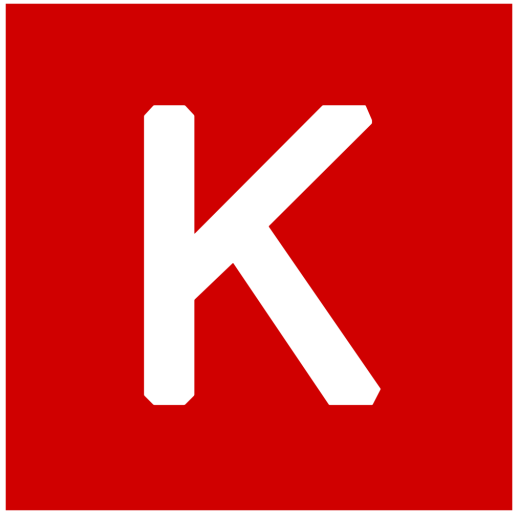
- 머신러닝 라이브러리 : scikit-learn(사이킷 런)

- DL Library(대표적)

• 딥러닝 라이브러리 : TensorFlow(Keras), PyTorch

TensorFlow

PyTorch



현재(2023년) 기준 PyTorch가 가장 인기 있음

- Scikit-learn, Keras model 비교

사이킷런 모델

손실 함수 ↓ 반복 횟수 ↓

```
모델 ⟶ sc = SGDClassifier(loss='log', max_iter=5)
훈련 ⟶ sc.fit(train_scaled, train_target)
평가 ⟶ sc.score(val_scaled, val_target)
```

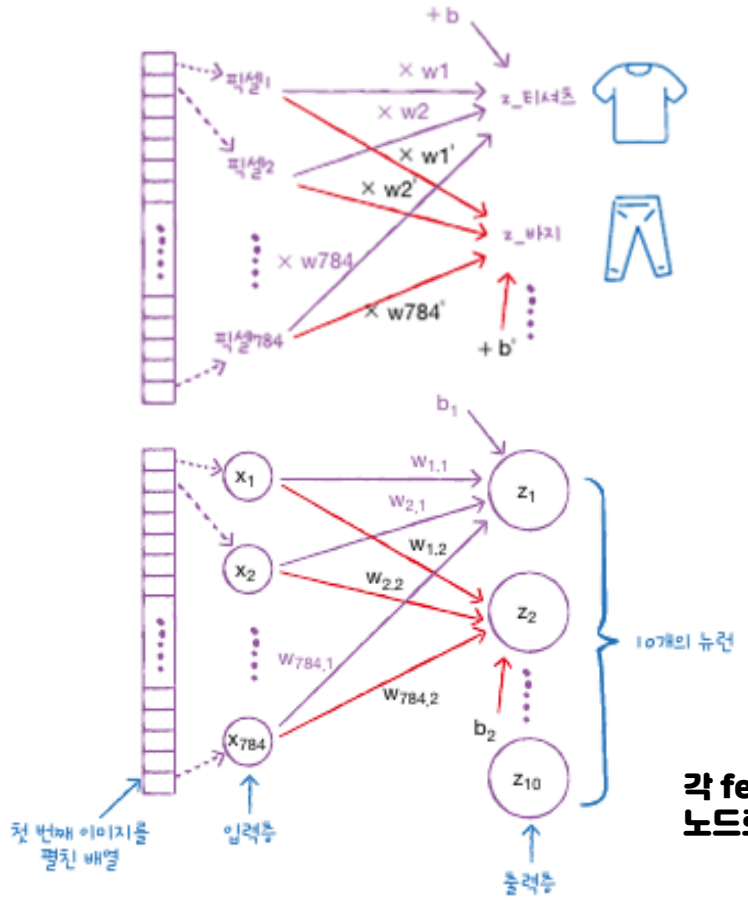
케라스 모델

층 생성 ↓

```
        dense = keras.layers.Dense(10, activation='softmax', input_shape=(784,))
모델 ⟶ model = keras.Sequential(dense)
        model.compile(loss='sparse_categorical_crossentropy', metrics='accuracy')
훈련 ⟶ model.fit(train_scaled, train_target, epochs=5)
평가 ⟶ model.evaluate(val_scaled, val_target)
```

손실 함수 ↓ (sparse_categorical_crossentropy)

반복 횟수 ↑ (epochs=5)

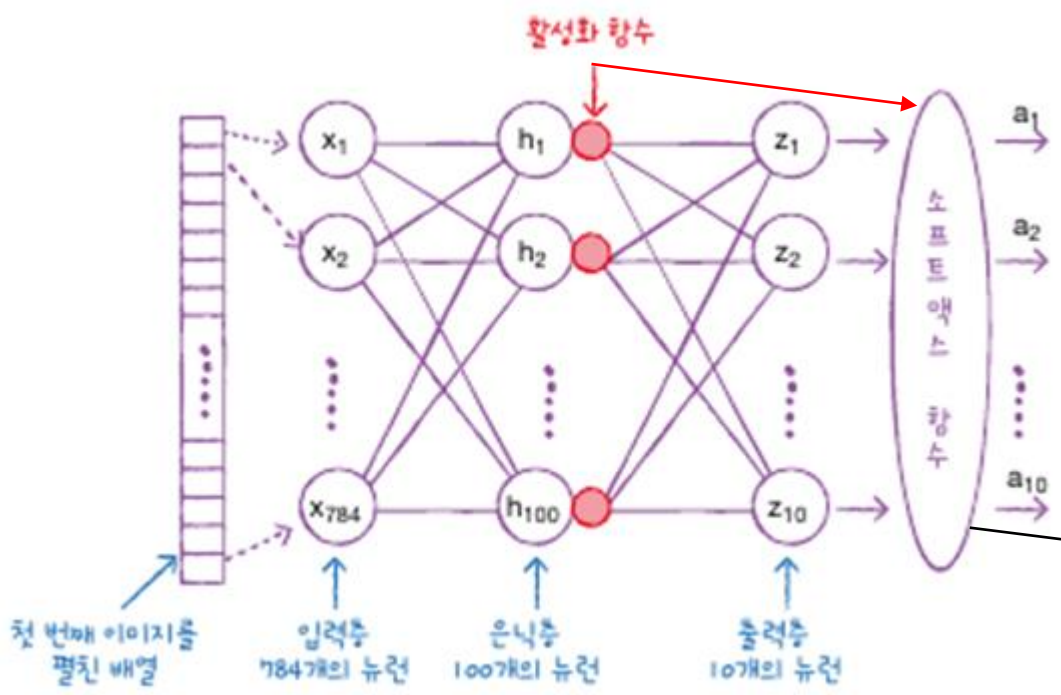


각 feature 및 output을 노드로 봄

Keras로 만든 위의 NN 모델은 사실상 scikit-learn으로 만든 모델과 거의 비슷
하지만 몇 가지 장점 덕분에 keras 모델이 조금 더 높은 성능을 냄

2. Deep Neural Network(심층 신경망) = Deep Learning

Activation Function : Neural Network Layer의 선형 방정식의 계산 값에 적용하는 함수



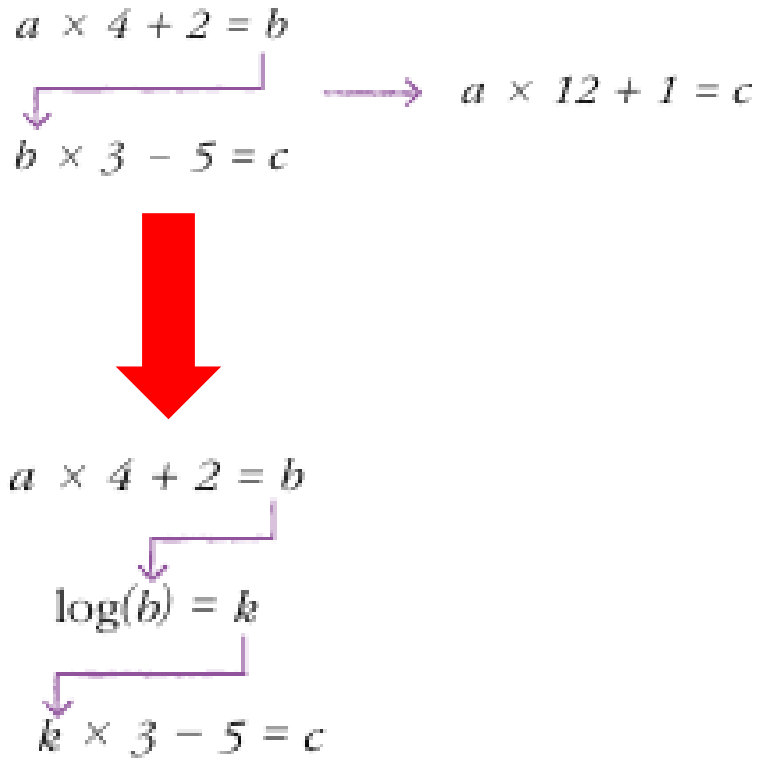
NN을 그림으로 나타낼 때,
bias(절편)과 activation function를 생략하는 경우가 많지만,
모든 hidden layer에는 bias와 activation function이 있음
(output layer도 마찬가지)

Regression의 output은 임의의 어떤 숫자이므로
activation function을 적용할 필요가 없음

즉, **output layer의 선형 방정식의 계산을 그대로 출력**

Hidden Layer : Input Layer와 Output Layer 사이에 있는 모든 층

심화! Hidden Layer에 Activation Function을 사용하는 이유

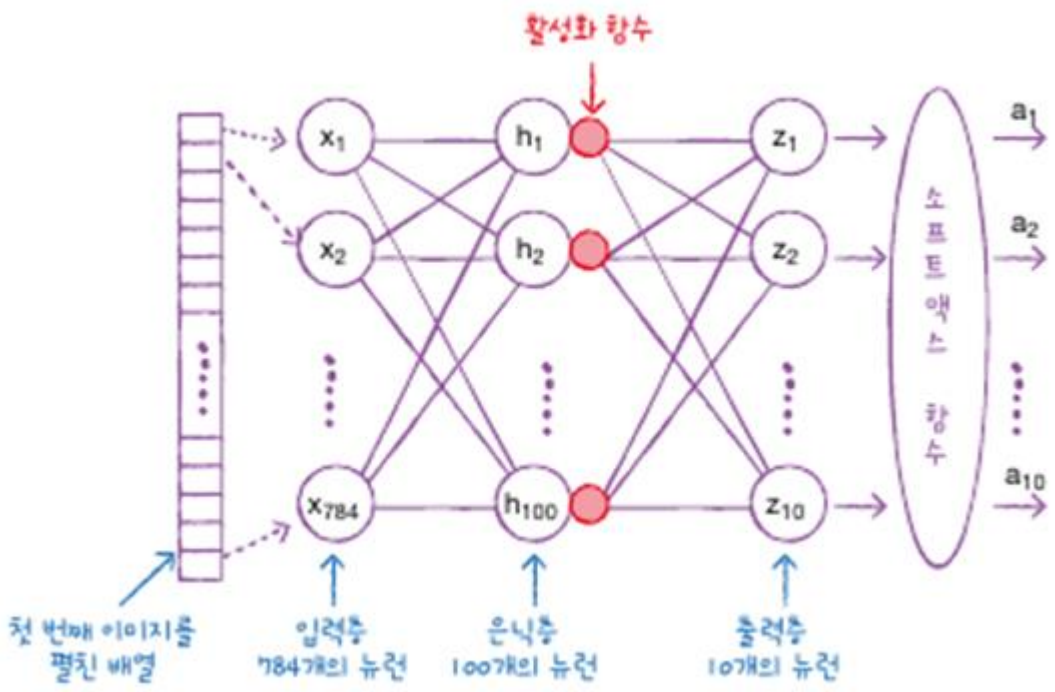


2개의 선형 방정식이 있을 때, 두 식을 하나로 합칠 수 있음
=> b가 사라지는 효과, b가 하는 일이 없음

선형 계산을 적당하게 비선형적으로 비틀어주어야
단순히 합쳐지는 것이 아니라 나름의 역할을 할 수 있음
=> b의 역할 존재

보통 hidden layer에는 sigmoid function을 많이 사용

심화! Model Summary



샘플 개수가 고정되어 있지 않으므로, None
이 배열의 첫번째 차원을 배치 차원이라 부름

Input Data
(None, 784)

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               78500
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1010
=================================================================
Total params: 79,510
Trainable params: 79,510
Non-trainable params: 0
_________________________________________________________________
```

MGD(미니배치 경사 하강법)를 사용해서 훈련

model parameter 수 = 이전 layer의 node 개수 x 현재 layer의 node 개수 + 현재 layer의 bias 수

78500 = 784 x 100 + 100
1010 = 100 x 10 + 10
=> 전체 model parameter = 78500 + 1010 = 79510

- Activation Function(활성화 함수)

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^{z}}{e^{z}+1}$$

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$$

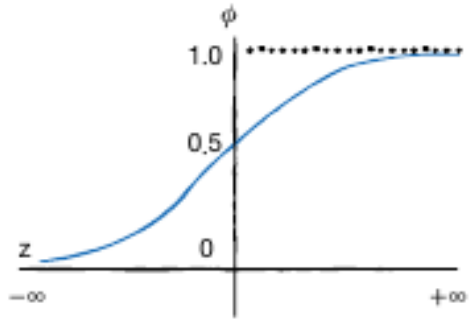
sigmoid 시그모이드

↓ 일반화 generalization

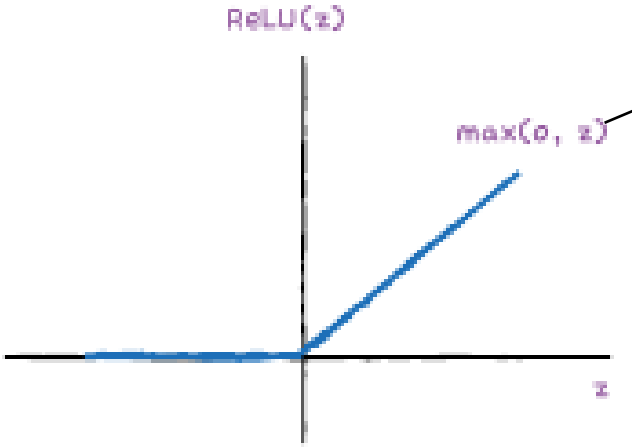
Softmax 소프트맥스

$$\phi = \frac{1}{1+e^{-z}}$$

시그모이드 함수



시그모이드 그래프



z가 0보다 크면 z를 출력
z가 0보다 작으면 0을 출력

ReLU Function은 이미지 처리에서 좋은 성능을 낸다고 알려져 있음

- Hyperparameter(하이퍼파라미터)

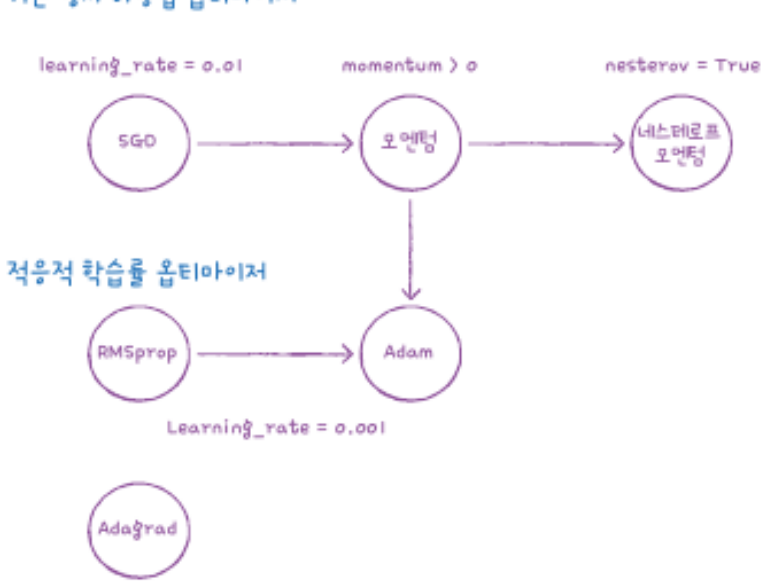
- NN에서의 hyperparameter
  - hidden layer 개수
  - node 개수
  - activation function
  - layer 종류(dense layer, …)
  - 배치 사이즈 매개변수(미니배치 경사 하강법)
  - 에포크 매개변수(전체 샘플 1회 훈련 = 1 에포크)
  - Optimizer(케라스의 기본 경사 하강법 알고리즘 : RMSprop)
  - Optimizer의 learning rate(학습률)
  - …

- Optimizer

- Optimizer in Keras
  - SGD(이름이 SGD이지만 미니배치를 사용)
  - Adagrad
  - RMSprop
  - Adam

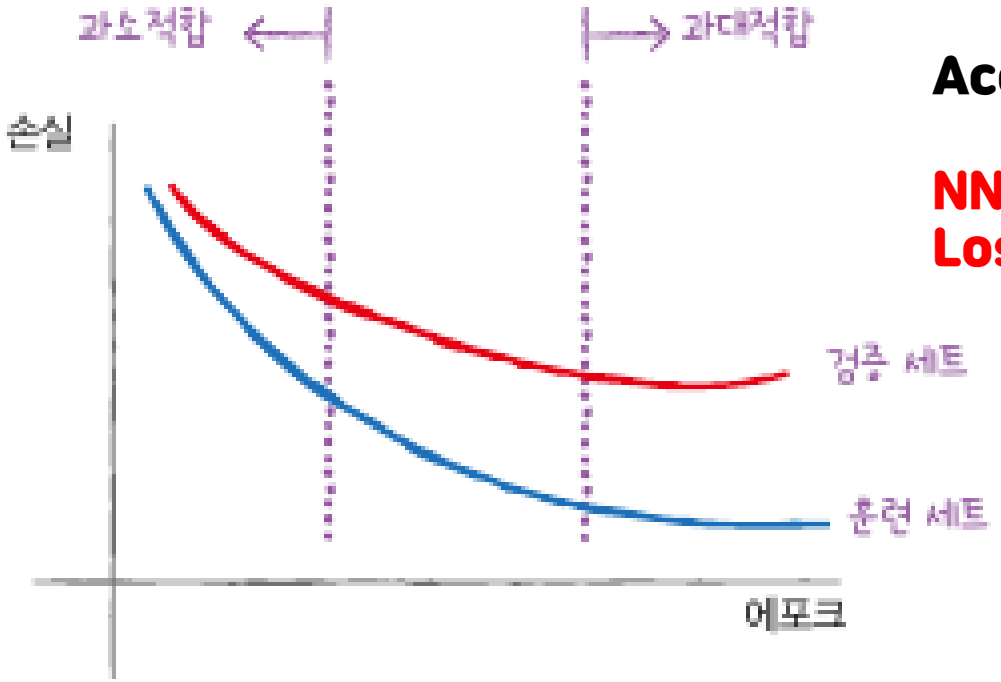
Adaptive Learning Rate(적응적 학습률) 사용하는 optimizer
- 모델이 최적점에 가까이 갈수록 학습률을 낮출 수 있음

Momentum optimization(모멘텀 최적화)와 RMSprop의 장점을 접목



RMSprop, Adam은 처음 시도하기에 좋은 알고리즘

- Overfitting/Underfitting

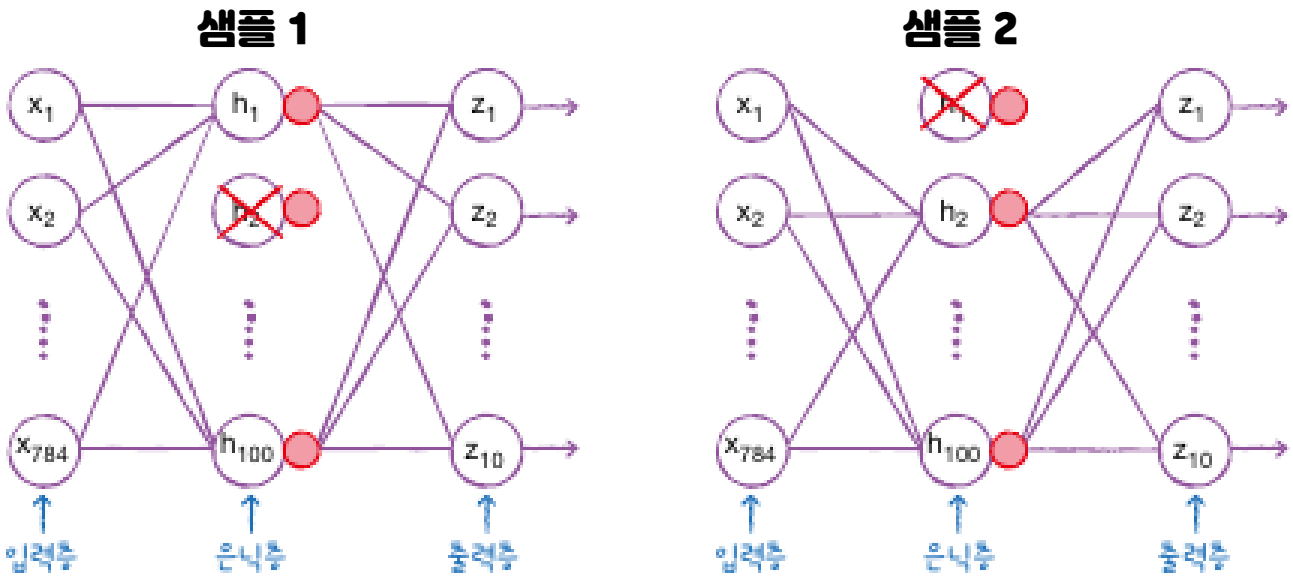


Accuracy로도 Overfitting/Underfitting을 따져볼 수 있음

NN 모델에서는 보통
Loss값으로 Overfitting/Underfitting을 따짐

NN 모델이 최적화하는 대상은 Loss function
- loss 감소에 비례하여 accuracy가 높아지지 않는 경우도 있음
- 모델이 잘 훈련되었는지 판단하려면 accuracy보다는 loss값을 확인하는 것이 더 나음

- Dropout(드롭아웃)



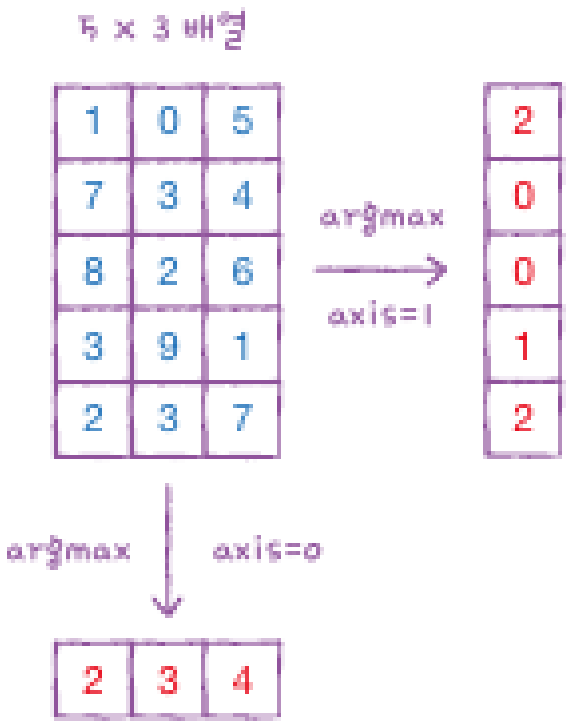
dropout을 하면 일부 노드의 출력을 0으로 만들지만
전체 출력 배열의 크기를 바꾸지는 않음

훈련이 끝난 뒤에 평가나 예측을 수행할 때는
dropout을 적용하면 안됨

훈련 과정에서 layer에 있는 일부 node를 랜덤하게 꺼서(즉 노드의 출력을 0으로 만들어) overfitting을 막음

샘플마다 꺼지는 node가 다르므로 일종의 앙상블로 생각할 수 있음 -> overfitting 방지

Tip! argmax() - axis



axis=0이면 행을 따라 각 열의 최댓값의 인덱스 선택,
axis=1이면 열을 따라 각 행의 최댓값의 인덱스 선택

axis=-1이면 배열의 마지막 차원을 따라 최댓값을 고름
2차원 배열일 경우 axis=1과 같음
3차원 배열일 경우 axis=2와 같음